Table of Contents

This document....2
- This document is not the 'Requirement specification'....2
- Proof-of-concept....2
- Feedback....2

Development Environment....3
Mediaserver-Application....4
- Constraints....4
- Modules....4
  - How to deploy and run....4
- Database....5
  - Prerequisite....5
  - Generating the tables....5
  - The 'ADMIN_CONFIG'-table : Generating configuration-content....5
  - The 'DETERMINATION'-table : Usage....6
  - Generating demo-data....6
- File-system....7

RESTful API : Basic Usage....8
- Prerequisite....8
- Support for standard CRUD-operations....8
  - Retrieve Mediafile : Valid for all Media-types....9
  - Retrieve a transformed Mediafile : Valid only for Images....9
  - Create....10
  - Update....11
  - Delete....12

RESTful API : Determination-linkage....13
- /determination/metadata/{extuuid}....13
- /determination/metadata/{extuuid}/{lang}....14
- /determination/metadata/{extuuid}/{lang}/{tags}....14
  - Send in the tag → view:flying....14
  - Send in the tag → view:sitting....14
  - Send in the tag → country:sweden....15

RESTful API : Upload batch....16
Demo user interface....17
- Prerequisite....17
  - Build&Deploy the following 2 Modules....17
- The Demo-data provided with the Mediaserver....17
- Test → http://localhost:8080/MockTaxonomyClient/....17

Glossary....18
Appendix A : Collected feedback....19
Appendix B : @ToDo....20
Appendix C : Defined Attributes....21

Organisation	Name	Changes	Version
NRM-BIO	Ingimar Erlingsson	First draft	1.0
NRM-BIO	Ingimar Erlingsson	Collceted-feedback (Appendix A)	1.1

This document

Is written for the Workshop in September 2014.

1. You are able to use the Mediaserver-Application both as a standalone server with no linkage to other systems.
2. You are able to use the Mediaserver-Application as an integrated part of your system, to manage media.

This document is not the 'Requirement specification'

That document can be found elsewhere. Can be provided.

Proof-of-concept

Integration with naturforskaren.se (not in prod as of 2014-09-17) Erlingsson&Skyttner.
Steps efore production:

- New import of all the images from naturforskaren.
  - SortOrder got fuzzed
- Cosmetic, CC-license-icons below
- Test the integration in the Test-Environment.
  - Merge the changes from naturforskaren that has been done since April 2014
  - Regression-tests.
    - The user-interface
    - The 'content provider'-interface

Feedback

Do you have any suggestions, improvements.
Please mail to : ingimar.erlingsson@nrm.se

Development Environment

The below has been used in this project.

System	Component	Version
OS	Linux/Ubuntu 64bit -desktop	14.04 LTS
RDBMS	MySQL	5.5.x
Application Server	Glassfish	3.1.2.2
Java Platform	Java SE	1.7 (1.7.0_51)
	Java EE (JPA + JAX_RS +*more*)	6
IDE	NetBeans	8.0
Project management tool	Maven ('mvn -v')	3.2.1
Source code management	git-client	1.9.1
Source control of the database	liquibase	3.0.7

Mediaserver-Application

Maven is used as a project management tool, maven is used primarily to build&deploy.

Constraints

- GUI
  - This project is a standalone server with no requirements concerning a graphical end-user-interface. But : The project has a minimal&ugly user-interface which posts its form-data to a RESTful-service.
- History-functionality
  - No history-functionality has been build into the system.
  - Not been integrated with any kind of 'user-module'

Modules

Based on the JavaEE6-stack.
Web-service oriented, using RESTful architectural style.

The project contains 4 modules, maven is used to bind the modules.

1. '**Web**'
   1. A module that exposes the RESTful-API (JAX-RS)
      1. Implemented with RESTEasy (http://resteasy.jboss.org/)
   2. Uses RestfulUtil-module, this module binds to via JNDI to the EJB.

2. '**EJB**'
   1. Connection to the database is done here, using JPA.

3. '**Common**' : Used by the 'Web-module' and the 'EJB-module'
   1. A module which holds the common entity-objects.

Modules	**Java Platform**	
Common	Java SE 1.7	JAR-file
Web-module	Java EE 6 Web + Java SE 1.7	WAR-file
RestfulUtil	Java SE 1.7	JAR-File
EJB	Java EE 6	JAR-file

How to deploy and run

Deploy the EJB-module and Web-module in the Application-Server.

Database

Note : In this example the RDBMS MySQL is used. Liquibase supports many other RDBMS.

Prerequisite

Before you run the below project you should:

1. Set up a MySQL-user and a password.
2. Create a database in the RDBMS.

Binding the database to the liquibase-project is done in the file liquibase.properties.

Generating the tables

The project contains a 'database-management'-module.
This is a liquibase-project to manage the mediaserver-database.
DML and DDL is generated when running this project.
All changes to the database is handled in on or multiple changelog-file(s), which is XML-style.

The 'ADMIN_CONFIG'-table : Generating configuration-content

Obs: No user-interface has been built to administer this table

The table has a classic key/value-approach.

With the ability to set the environment ; ie. 'development' / 'test-environment' / 'production'.

```
mysql> desc ADMIN_CONFIG;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| ID          | int(11)      | NO   | PRI | NULL    | auto_increment |
| environment | varchar(255) | YES  |     | NULL    |                |
| admin_key   | varchar(255) | YES  |     | NULL    |                |
| admin_value | varchar(255) | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

The content is provided from the liquibase-file 'default-insert-for-admin_config.sql':

1. INSERT INTO `nrm_media`.`ADMIN_CONFIG`
   1. VALUES (null,'development', 'is_exif', 'false');
2. INSERT INTO `nrm_media`.`ADMIN_CONFIG`
   1. VALUES (null,'development', **'path_to_files', '/opt/data/mediaserver/newmedia/');**
3. INSERT INTO `nrm_media`.`ADMIN_CONFIG`
   1. VALUES (null,'development', **'base_url', '/MediaServerResteasy/');**
4. INSERT INTO `nrm_media`.`ADMIN_CONFIG`
   1. VALUES (null,'development', **'relative_metadata_url', 'media/image/');**
5. INSERT INTO `nrm_media`.`ADMIN_CONFIG`
   1. VALUES (null,'development', **'relative_stream_url', 'media/stream/');**

The 'DETERMINATION'-table : Usage

The table is used to **link** to an external-application to the mediaserver.
The table has a classic key/value-approach.

For instance to link one Taxonomy-UUID to many Media-UUID:s.

Proof-of-concept:

- obs → not in production-environment
- The Integration of NRM:s www.naturforskaren.se and the Mediaserver

Generating demo-data

Demo-data (meta-data for #Media-objects) is included.
A UUID is used as the identifier for a the Media-objects metadata-
The Media-objects file is identified by that same UUID.

File-system

You have to decide where your filesystem the mediafiles should reside and create that directory.
Update the database table *ADMIN_CONFIG*: the key has to be **'path_to_files'**

- **In our example : 'path_to_files', '/opt/data/mediaserver/newmedia/'**

Create the directory → /opt/data/mediaserver/newmedia/ .
Remember to set the properties for read-and-write on the directory.

The media-files that you upload to the Mediaserver is stored under that directory, in a sub-directory. The media-file is given a new name, a name that is a UUID created by the Mediaserver-Application. The media-file is stored as is, it is stored in its raw-file – the file is not manipulatet.

The principle is the following :

    if the generated UUID is ' ab30899c-58a0-4305-85a6-bbfa14f89b92'

Then the file is stored in the following directory (subdirectory is made up by the first 3 chars):

    /opt/data/mediaserver/newmedia/a/b/3

RESTful API : Basic Usage

Supported Media-types are today

1. Images
2. Streams of
   1. Video-files
   2. Sound-files
3. Attachment
   1. ie. PDF

OBS! Some constraints has been set on the supported MIME-type **– for now.**

Prerequisite

- Deploy the Mediaserver-Application
  - in our example **Server** = http://locahost:8080
- Check that the demo-data is present : in the database and in the filesystem
- Using UUID = **863ec044-17cf-4c87-81cc-783ab13230ae**

Support for standard CRUD-operations

OBS!:

1. XML-style : The meta-data is delivered in
2. JSON-style : Implemented , but not fully tested.
3. Versioning the URI is on the schedule

OBS!: Do not try to retrieve the media-file from the terminal, use the browser.

Tool: cURL(http://curl.haxx.se/) using cURL to call the RESTful-webservices.

See Appendix C for supported attributes.

Tagging :Generic Tags are supported, key/value, and saved as a text-string (see example #2)

1. '&' is delimiter between key:value
2. i.e : country:sweden&test:true

? Audubon and Darwin-core,

Saving the mdhash for every media-file , if you would like to find duplicates in the future.
Support for start-time and end-time for streaming-files (Video-files and Sound-files)

Retrieve Mediafile : Valid for all Media-types

Where '**Server**' = http://localhost:8080

1. Retrieve the Mediafile **meta-data**.
    1. **http://localhost:8080**/MediaServerResteasy/media/**metadata**/863ec044-17cf-4c87-81cc-783ab13230ae


```
▼<image>
   <uuid>863ec044-17cf-4c87-81cc-783ab13230ae</uuid>
   <owner>ingimar</owner>
   <visibility>public</visibility>
   <mimetype>image/jpeg</mimetype>
 ▼<mediaURL>
     http://localhost:8080/MediaServerResteasy/media/stream/863ec044-17cf-4c87-81cc-783ab1
   </mediaURL>
 ▼<taggar>
     view:sitting&country:sweden&photo:diginatour&source:wiki
   </taggar>
 ▼<tags>
   ▼<tag>
```


2. Retrieve the actual **Mediafileobject**
    1. http://localhost:8080/MediaServerResteasy/media/**stream**/863ec044-17cf-4c87-81cc-783ab13230ae

Retrieve a transformed Mediafile : Valid only for Images.

Transformation on-the-fly.
The API used for transformation is the 'thumbnailator' from net.coobird (http://coobird.net/).
'thumbnailator' keeps the proportion of the image.

OBS! A lower boundry has been set to **height= 100, width = 100** (hardcoded)

The structure of the RESTful-URI → ("/stream/image/{uuid}/{height}/{width}")

1. **height= 200, width = 200** ('thumbnailator' gives height= 200, width = 133)
    1. **http://localhost:8080**/MediaServerResteasy/media/**stream/image/**863ec044-17cf-4c87-81cc-783ab13230ae/200/200
2. **height= 300, width = 300** ('thumbnailator' gives height= 259, width = 173)
    1. **http://localhost:8080**/MediaServerResteasy/media/stream/image/863ec044-17cf-4c87-81cc-783ab13230ae/300/300

Create

Mind the syntax: when posing a file with cURL , you have to put the '@'-sign in front of the file

URI: '/MediaServerResteasy/media/upload-file'

Method = Post:

→ curl -v -F "owner=Stefan Daume" -F "access=public" -F "licenseType=CC BY" -F "legend=en skata" -F "legendLanguage=sv_SE" -F "tags=view:left" -F "fileName=pica-pica-flying.jpg" -F "**selectedFile=@pica-pica-flying.jpg**" http://127.0.0.1:8080/MediaServerResteasy/media/upload-file

Response if HTTP '200 OK':

→ <UUID>

for instance, in this case→ 46853e82-6cad-430b-b582-90e85203dce8

Test :

→ curl http://localhost:8080/MediaServerResteasy/media/metadata/<UUID>

for instance, in this case→ 46853e82-6cad-430b-b582-90e85203dce8

→ curl http://localhost:8080/MediaServerResteasy/media/metadata/ 46853e82-6cad-430b-b582-90e85203dce8

Note:

Sound :

curl -v -F "owner=Stefan Daume" -F "access=public" -F "licenseType=CC BY" -F "legend=Gräshoppa" -F "legendLanguage=sv_SE" -F "tags=sound:nature" -F "fileName=grasshopper.ogg" -F "selectedFile=@grasshopper.ogg" http://127.0.0.1:8080/MediaServerResteasy/media/upload-file

Video :

curl -v -F "owner=Stefan Daume" -F "access=public" -F "licenseType=CC BY" -F "legend=Foo Fighters" -F "legendLanguage=sv_SE" -F "tags=music:pop" -F "fileName=Fighters.mp4" -F "selectedFile=@video.mp4" http://127.0.0.1:8080/MediaServerResteasy/media/upload-file

Update

URI: '/MediaServerResteasy/media/upload-file'

ex: Change owner to Karin Karlsson

Method = PUT:

→ curl -X **PUT** -F "mediaUUID=46853e82-6cad-430b-b582-90e85203dce8" -F "owner=Karin Karlsson" http://127.0.0.1:8080/MediaServerResteasy/media/upload-file

Retrieve:

-work do be done

Test :

→ curl http://localhost:8080/MediaServerResteasy/media/metadata/<UUID>

for instance, in this case→ 46853e82-6cad-430b-b582-90e85203dce8

→ curl http://localhost:8080/MediaServerResteasy/media/metadata/46853e82-6cad-430b-b582-90e85203dce8

Delete

Today : The meta-data and the file is erased, would it be better to 'disable' the meta-data and the file ?

Two URI:s are provided, ajax-techonolgy must use @GET (cannot use @DELETE)

1. @GET
   1. @Path("/ajax/delete/all/{mediaUUID}")
2. @DELETE
   1. @Path("/delete/all/{mediaUUID}")

Prerequisite for test: Check if the file is there!

1. List the file.
2. ingimar@ingimar-HP-EliteBook-8470p:~$ **ll** /opt/data/mediaserver/newmedia/4/6/8/

Prerequisite for test: Check if the metadata is there!

1. curl http://localhost:8080/MediaServerResteasy/media/metadata/468b5d54-9026-48be-b174-a088ddb3c0c8

URI: '/MediaServerResteasy/media/delete/all/<UUID>'

Method = DELETE:

→ curl -i -X DELETE http://localhost:8080/MediaServerResteasy/media/delete/all/46853e82-6cad-430b-b582-90e85203dce8

Response:

→ Boolean : true or false

Test:

Check if the file is there!

1. List the file.
2. ingimar@ingimar-HP-EliteBook-8470p:~$ **ll** /opt/data/mediaserver/newmedia/4/6/8/

Check if the metadata is there!

1. curl http://localhost:8080/MediaServerResteasy/media/metadata/468b5d54-9026-48be-b174-a088ddb3c0c8

RESTful API : Determination-linkage

Example:

1. Pica Pica (Skata) with TAG_KEY= '02d2ef271-02ca-4934-860c-6c6a4ed043f9'
2. There are 2 images stored.
   1. **'Skata' is flying:**
      1. tagged → `view:flying&country:sweden`
      2. /MediaServerResteasy/media/stream/c41bd445-8796-4421-9b77-fd1e65b14974
      3. <descriptions><description><uuid>10158</uuid><legend>Skatan flyger! </legend>**<lang>sv_SE</lang>**</description></descriptions>
   2. **'Skata' is sitting:**
      1. tagged → `view:sitting&country:sweden&photo:diginatour&source:wiki`
      2. /MediaServerResteasy/media/stream/863ec044-17cf-4c87-81cc-783ab13230ae
      3. <descriptions><description><uuid>1</uuid><legend>beskrivning av skatan</legend>**<lang>sv_SE</lang>**</description></descriptions>

URI:s

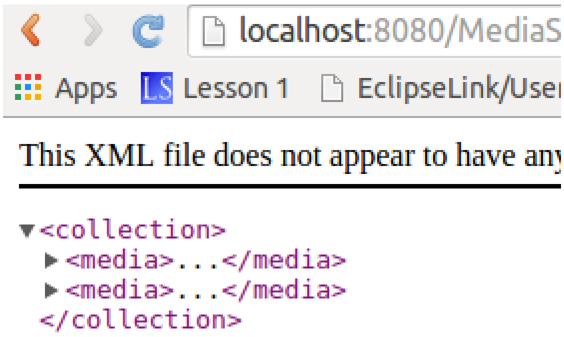
1. @Path("/determination/metadata/{extuuid}")
2. @Path("/determination/metadata/{extuuid}/{lang}")
3. @Path("/determination/metadata/{extuuid}/{lang}/{tags}")
4. @Path("/determination/present/{extuuid}")
   1. returns a Boolean.

/determination/metadata/{extuuid}

Request:

→ **'server'**/MediaServerResteasy/media/determination/metadata/02d2ef271-02ca-4934-860c-6c6a4ed043f9

Response:

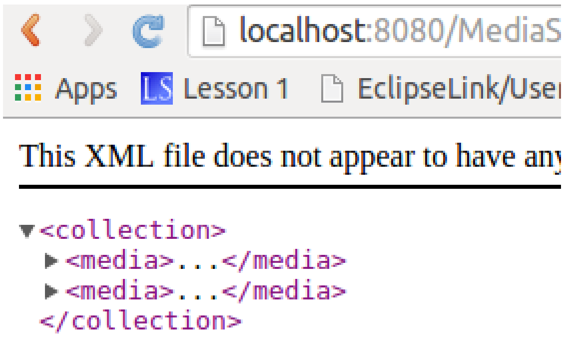


/determination/metadata/{extuuid}/{lang}

Request:

→ **'server'**/MediaServerResteasy/media/determination/metadata/02d2ef271-02ca-4934-860c-6c6a4ed043f9/sv_SE

Response: Same response



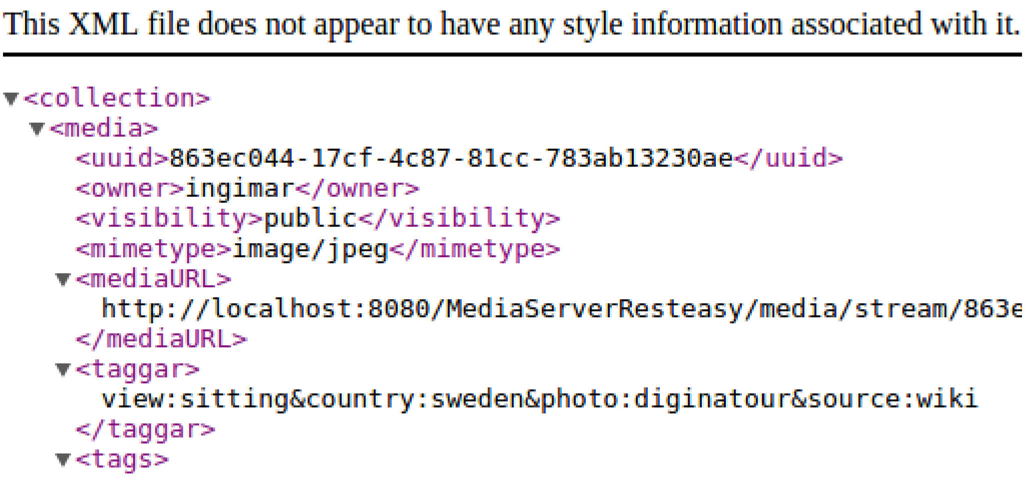
/determination/metadata/{extuuid}/{lang}/{tags}

Send in the tag → view:flying

http://localhost:8080/MediaServerResteasy/media/determination/metadata/02d2ef271-02ca-4934-860c-6c6a4ed043f9/sv_SE/view:flying

Send in the tag → view:sitting

http://localhost:8080/MediaServerResteasy/media/determination/metadata/02d2ef271-02ca-4934-860c-6c6a4ed043f9/sv_SE/view:sitting



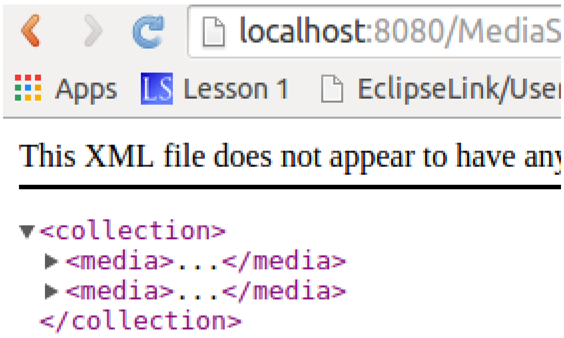
This XML file does not appear to have any style information associated with it.


```
▼<collection>
  ▼<media>
     <uuid>863ec044-17cf-4c87-81cc-783ab13230ae</uuid>
     <owner>ingimar</owner>
     <visibility>public</visibility>
     <mimetype>image/jpeg</mimetype>
    ▼<mediaURL>
       http://localhost:8080/MediaServerResteasy/media/stream/863e
     </mediaURL>
    ▼<taggar>
       view:sitting&country:sweden&photo:diginatour&source:wiki
     </taggar>
    ▼<tags>
```


Send in the tag → country:sweden

http://localhost:8080/MediaServerResteasy/media/determination/metadata/02d2ef271-02ca-4934-860c-6c6a4ed043f9/sv_SE/country:sweden

Reponse:



This XML file does not appear to have any


```
▼<collection>
  ▶<media>...</media>
  ▶<media>...</media>
 </collection>
```

RESTful API : Upload batch

Proof-of-concept : All the media-files and their meta-data from naturforskaren.se

Example from naturforskaren : Everything should be typed in on one row

- curl -i -X POST
- **-H 'Content-Type: application/json'**
- -d '{"typeOfSystem":"NF_TAXON","taxonUUID":"taxon-nr","nameOfSystem":"NF_SYSTEM","systemURL":"NF_URL","**mediaList**":["e4a3cf7d-add4-4949-a6ce-0f5594e61970","ebb45da5-bd25-45af-8a04-3470d38523d1"]}'
- http://172.16.23.62:8080/MediaServerResteasy/media/postJSONWithLIsta

TWO RESTful-URI:s (*due to wrongspelling of the first URI*), perform the same job.

1. @Path("/upload-batch-coupling") @Consumes({MediaType.APPLICATION_JSON})
   1. Calls → "/postJSONWithLIsta"

2. @Path("/postJSONWithLIsta") @Consumes({MediaType.APPLICATION_JSON})
   1. Wrong spelling, that is why there is a better name above.
   2. This method does the actual job.

Demo user interface

1. The demo-data without an external system
   1. Fetch the Image (UUID = 863ec044-17cf-4c87-81cc-783ab13230ae)
   2. Fetch the transformed image
2. The demo-data with an external system (Dummy-taxonomy-Database)
   1. Ability to search for 'skata' and one unique tag.
   2. 'skata' has the Dummy-taxonomy-UUID = '02d2ef271-02ca-4934-860c-6c6a4ed043f9'

Prerequisite

1. Start the Mediaserver-Application.
2. Check that there is a TaxonMock-db with content

Build&Deploy the following 2 Modules

1. SimpleTaxonMock
2. WebMockTaxon

Dummy Taxonomy Database
1. Database : TaxonMock
2. Table: mock_taxon

```
mysql> select * from mock_taxon;
+----+--------------------------------------+-------------+-----------------+
| id | ext_uuid                             | common_name | scientific_name |
+----+--------------------------------------+-------------+-----------------+
|  1 | 02d2ef271-02ca-4934-860c-6c6a4ed043f9 | skata       | Pica pica       |
|  2 | 1089d789a-2df5-4882-abce-76d4f22e0a7a | Korp        | Corvus corax    |
|  4 | fe1b99c3-d517-4ca0-a5e5-a2545e59c9a2  | Människa    | Homo Sapiens    |
|  6 | b0b7e712-7d1b-46a7-985a-b7913e07d58d  | duper       | super           |
|  7 | f53f2bd4-fa72-4fdc-a136-9873cae4b48b  | fish fish   | cod             |
+----+--------------------------------------+-------------+-----------------+
5 rows in set (0.00 sec)
```

The Demo-data provided with the Mediaserver

'skata' is one of the items in the demo-data provided with the Mediaserver.
Verify that with the following sql-statement.
mysql> select * from DETERMINATION where TAG_VALUE='02d2ef271-02ca-4934-860c-6c6a4ed043f9';

Test → http://localhost:8080/MockTaxonomyClient/

Test-1
1. Request : name = skata ,tags = view:sitting
2. Response : an image of a sitting 'skata'

Test-2
1. Request : name = skata ,tags = view:flying
2. Response : an image of a flying 'skata

Glossary

Term		URI
JEE6	Java Enterprise Edition , version 6	http://docs.oracle.com/javaee/6/tutorial/doc/
Java SE	Java Standard Edition	
Maven	“Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.”	http://maven.apache.org/
Liquibase	Database refactoring/Source control	http://www.liquibase.org/
DML	Data Manipulation Language	
DDL	Data Definition Language	
UUID	Universally unique identifier, 128-bit number.	http://tools.ietf.org/html/rfc4122
RESTful	“The REST architectural style is also applied to the development of web services as an *alternative* to other distributed-computing specifications such as SOAP”	http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
cURL	Tool	http://curl.haxx.se/
ajax		http://en.wikipedia.org/wiki/Ajax_(programming)

Appendix A : Collected feedback.

1. About the XML (Guido)
    1. Why not use 'attributes' instead of using containers – easier to parse with that approach
2. Licenses (Woutar)
    1. More information is needed.
    2. See the handling in 'http://artsdatabanken.no/' ,
        1. See → http://artsdatabanken.no/Article/Article/134019
            1. See → http://artsdatabanken.no/Files/1733,1731
3. About the use of an application server, is that necessary
    1. making it more lean – look into that.

Appendix B : @ToDo

This is an excerpt from the requirement-specification and discussion:

1. Ability to 'dump' all EXIF-data.
    1. Switch in the ADMIN_CONFIG : boolean
2. mark for export
    1. Switch in the ADMIN_CONFIG : boolean
3. tag multiple taxon in one image (an idea that might not be realised)
4. tagging a media-object is done via a generic tagging functionality
    1. anyone can create their own keys.
        1. This has a downside
    2. The creator of an Application that you want to integrate with the mediaserver
        1. you can restrict the tags in the useri-interface (# of audobon) and save them in the 'tags' field.
5. Delete content or Disable content ?
6. Transformation of images
    1. On the Fly – Done
    2. When saving an image : Save the raw form + thumbnail + 'medium'-dimension
    3. instead of the URI → /<UUID>/height/width
        1. Use → /<UUID>/thumbnail
        2. Use → /<UUID>/medium
        3. Use → /<UUID>/orginal
7. Versioning the URI is on the schedule
8. And a couple of more ….

Appendix C : Defined Attributes

Fields that are available (Module 'Web' class FileUploadForm.java)

The below is an excerp from the java-class that handles the attributes.

```
@FormParam("mediaUUID")
@FormParam("selectedFile")
@PartType("application/octet-stream")
@FormParam("owner")
/**
 * i.e: private or public
 */
@FormParam("access")
/**
 * '&' is delimiter between key:value
 *
 * i.e : country:sweden&test:true
 */
@FormParam("tags")
@FormParam("legend")
@FormParam("legendLanguage")
@FormParam("licenseType")
@FormParam("fileName")
@FormParam("alt")
@FormParam("exportImage")
@FormParam("startTime")
@FormParam("endTime")
@FormParam("comment")
@FormParam("displayOrder")
```